

Charleston Java User Group

An EJB 3 Primer

Reza Rahman

Author, *EJB 3 in Action*

Expert Group Member, Java EE 6 and EJB 3.1

Founder, Cognicellence

October, 2008

EJB: A Mixed Past

- **EJB is the pioneer Java server-side enterprise component model**
- **Inspired by CORBA, early focus was on distributed computing**
- **Brought component services to the mainstream**
 - **Transactions**
 - **Security**
 - **Persistence**
 - **Messaging**
 - **Web services/remoting**
 - **Threading, pooling, component state and life-cycle management**
- **Although theoretically compelling, previous versions of EJB have been widely criticized for complexity**
 - **Redundant, Java inheritance based code**
 - **Heavy XML configuration with deployment descriptors**
 - **Flawed, half-baked Entity Beans persistence model**
 - **Too many things left up to vendors**
- **Spring (and PicoContainer) + Hibernate (and JDO, TopLink)**
 - **POJO programming**
 - **Dependency injection, aspect-oriented programming**

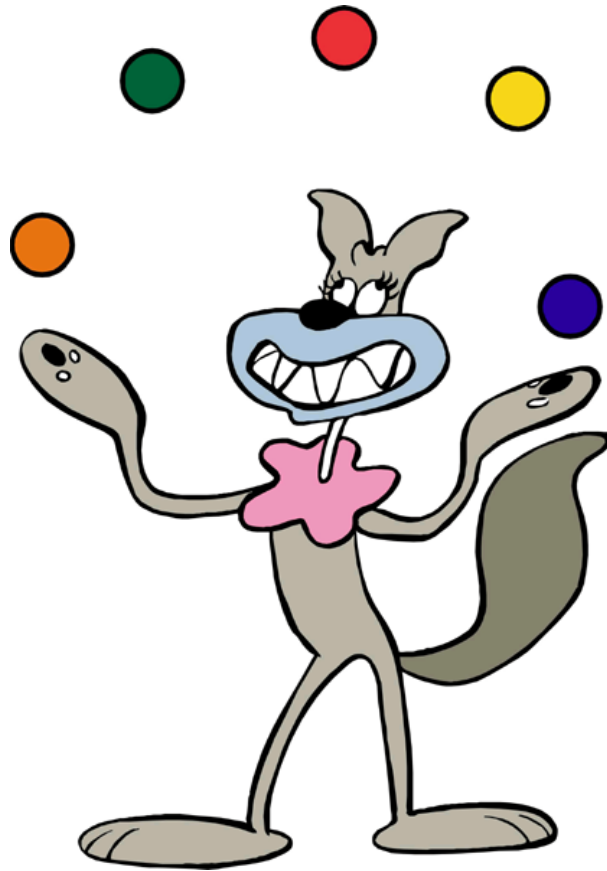
EJB 3: A Radical Reinvention

- **EJB 3 remakes the technology for a new generation of Java EE developers**
- **A radical, futuristic overhaul**
 - *POJO* programming
 - *100% annotations*
 - XML deployment descriptors optional, can be used as an override mechanism
 - *Intelligent defaulting* whenever possible, similar to *convention over configuration* in Rails
 - Entity Beans scrapped
 - Adopts the *Java Persistence API (JPA)*, an API paradigm similar to Hibernate, TopLink and JDO.
 - Maximum portability, including relational mapping and object queries
 - Injection and interception
- **Changes make EJB 3 a compelling choice on its own right**
- **Just may be the easiest, most powerful server-side component development model to work with**
- **Integrates with Seam**
- **Can be used with embedded EJB 3 containers and Tomcat**

The Anatomy of EJB 3

- **Session Bean - general-purpose component**
 - Stateless Session Bean - does not store state
 - Stateful Session Beans - guaranteed to store state
- **Message-Driven Beans - processes asynchronous messages**
- **Java Persistence API - persistence using Object Relational Mapping (ORM)**
 - Entities - persistent objects that are stored in a relational database
 - EntityManager - API for performing persistence operations (adding, updating, deleting, retrieving entities)
 - Java Persistence Query Language (JPQL): SQL-like language used to retrieve and manipulate entities

The Demo!



Some More EJB 3 Features

- **Session Beans**
 - Stateful Beans
 - Passivation
 - Life-cycle management
 - Security
 - Timers
 - Thread management, pooling, clustering
- **Message-Driven Beans**
 - Life-cycle management
 - JMS topics
 - Message acknowledgement, durability, filtering
 - Thread management, pooling, clustering
- **JPA**
 - One-one, many-many relations
 - Mapping inheritance
 - JPQL joins
 - JPQL ordering, grouping and summarization
 - Subqueries
 - Native queries

The Changes in EJB 3.1

- **Session Bean optional interfaces**
- **Singleton Beans with concurrency control**
- **Cron-style declarative and programmatic Timers**
- **Asynchronous bean invocation**
- **Simplified WAR packaging**
- **Java SE support**
- **Standardized Global JNDI naming**
- **EJB Lite**

Cron-like Declarative Timers

`@Stateless`

```
public class NewsLetterGeneratorBean {  
    @Resource  
    private Session mailSession;  
  
    @Schedule(second="0", minute="0", hour="0",  
              dayOfMonth="1", month="*", year="*")  
    public void generateMonthlyNewsLetter() {  
        ...  
    }  
}
```

Asynchronous Session Bean

`@Stateless`

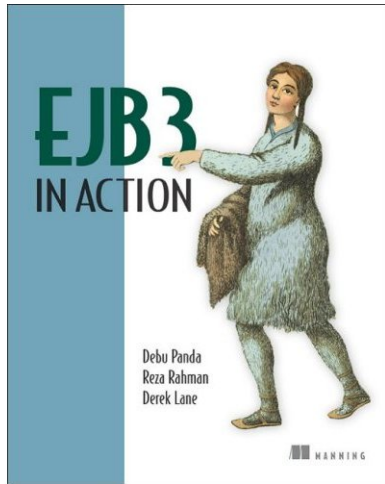
```
public class OrderBillingBean {  
    ...  
    @Asynchronous  
    public Future<BillingStatus> billOrder(Order order) {  
        try {  
            bill(order);  
            return new AsyncResult<BillingStatus>(  
                BillingStatus.COMPLETE);  
        } catch (BillingException be) {  
            return new AsyncResult<BillingStatus>(  
                BillingStatus.BILLING_FAILED);  
        }  
    }  
    ...  
}
```

Asynchronous Invocation Client

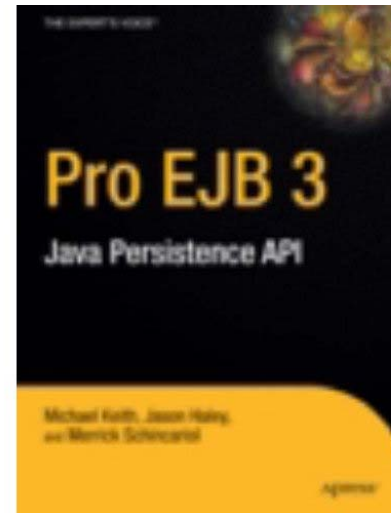
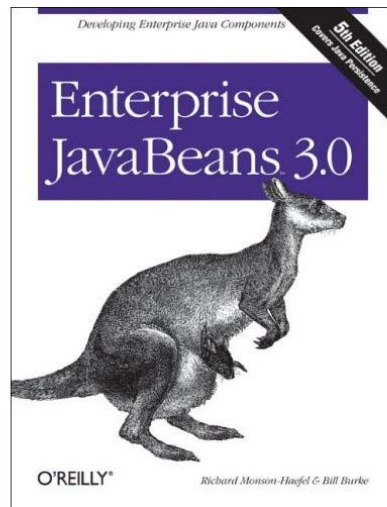
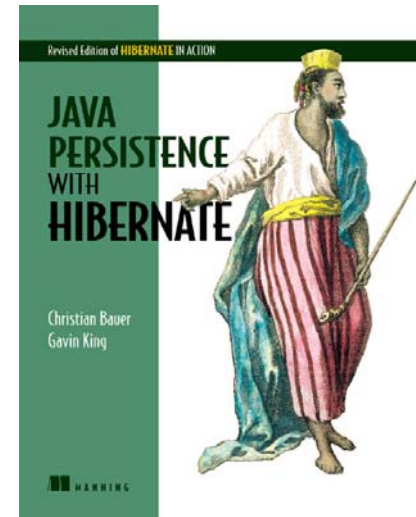
@EJB

```
private OrderBillingBean orderBilling;
...
Order order = new Order();
...
Future<BillingStatus> future = orderBilling.billOrder(order);
...
BillingStatus status = future.get();
...
if (status == BillingStatus.COMPLETE) {
    notifyBillingSuccess(order);
} else if (status == BillingStatus.BILLING_FAILED) {
    notifyBillingFailure(order);
}
```

References



Shameless plug alert!



Frequently Asked Questions

- **Can you use EJB 3 with Tomcat?**
 - Yes, you can do this using an embeddable container like *OpenEJB* or the *JBoss Embeddable Container*
- **How can you unit/integration test EJB 3?**
 - You can do this using manual JNDI lookups, an application client container, EasyGloss or an embeddable container
- **What containers support EJB 3?**
 - GlassFish, WebLogic, WebSphere, Oracle, JBoss
- **What JPA persistence providers are available?**
 - Hibernate, TopLink, Kodo, OpenJPA
- **Do I need Java 5 for EJB 3?**
 - In general, yes
- **Can I use EJB 3 with Spring?**
 - Yes, many integration possibilities exist